

Student name: Lê Ngọc Anh Vũ

Student ID: 20236014

Assignment 1: Create a new project to implement the program in Home Assignment 1. Compile and upload to simulator. Run and observe the result. Go to Data Segment, check how test string is stored and packed in memory.

The screenshot shows the RARS 1.6 assembly debugger interface. At the top, the assembly code for 'lab 5.asm' is displayed:

```
1 # Laboratory Exercise 5, Assignment 1
2 .data
3     test: .asciz "Hello World"
4 .text
5     li    a7, 4
6     la    a0, test
7     ecall
```

The Registers window on the right shows the following register values:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10010000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000

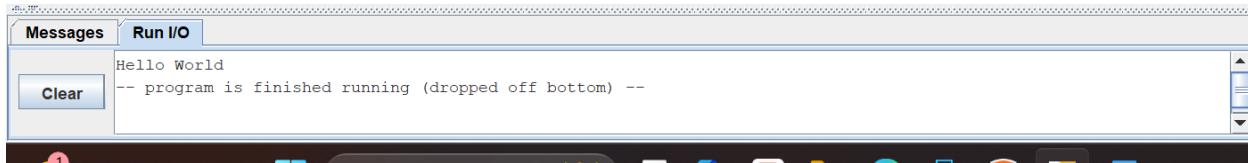
The Data Segment window shows the memory dump starting at address 0x10010000:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x6c6c6548	0x6f57206f	0x00646c72	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

The Run I/O window shows the output:

```
Hello World
-- program is finished running (dropped off bottom) --
```

After running the program, the Run I/O represents the test string "Hello World" as the picture below:



Display data segment values in ASCII.

Test string is stored and packed as the picture below:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00400893	addi x17,x0,4	5: li a7, 4
	0x00400004	0x0fc10517	auipc x10,0x0000fc10	6: la a0, test
	0x00400008	0xfc50513	addi x10,x10,0xfffffffffc	
	0x0040000c	0x00000073	ecall	7: ecall

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1 1 e H	o w o	\0 d 1 r	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Assignment 2: Create a new project to print the sum of two register \$s0 and \$s1 according to this format:

“The sum of (s0) and (s1) is (result)”

The source code is as below:

We assign the register s0 to 4, the register s1 to 1

We is going to print : “The sum of 4 and 1 is 5”

Edit Execute

lab 5.asm*

```
1 # Laboratory Exercise 5, Assignment 2
2 .data
3     mess1: .asciz "The sum of "
4     mess2: .asciz " and "
5     mess3: .asciz " is "
6 .text
7     addi $0, zero, 4
8     addi $1, zero, 1
9
10    li a7, 4
11    la a0, mess1
12    ecall
13
14    li a7, 1
15    add a0, $0, zero
16    ecall
17
18    li a7, 4
19    la a0, mess2
20    ecall
21
22    li a7, 1
```

Edit Execute

lab 5.asm*

```
12     ecall
13
14     li a7, 1
15     add a0, $0, zero
16     ecall
17
18     li a7, 4
19     la a0, mess2
20     ecall
21
22     li a7, 1
23     add a0, $1, zero
24     ecall
25
26     li a7, 4
27     la a0, mess3
28     ecall
29
30     li a7, 1
31     add a0, $0, $1
32     ecall
```

Line: 8 Column: 18 Show Line Numbers

The first ecall

The screenshot shows the RARS 1.6 debugger interface. The assembly code window displays the following instructions:

```

Text Segment
Bkpt Address Code Basic Source
0x00400000 0x00400413 addi x8,x0,4 7: addi s0, zero, 4
0x00400004 0x00100493 addi x9,x0,1 8: addi s1, zero, 1
0x00400008 0x00400893 addi x17,x0,4 10: li a7, 4
0x0040000c 0x0fc1c0517 auipc x10,0x0000fc10 11: la a0, messl
0x00400010 0xffff450513 addi x10,x10,0xffffffff4
0x00400014 0x00000073 ecall 12: ecall
0x00400018 0x00100893 addi x17,x0,1 14: li a7, 1
0x0040001c 0x00040533 add x10,x8,x0 15: add a0, s0, zero
0x00400020 0x00000073 ecall 16: ecall

```

The Registers window shows the following register values:

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000004
s1	9	0x00000001
a0	10	0x10010000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400018

The Data Segment window shows memory starting at address 0x10010000. The Messages window displays "The sum of 4".

The second ecall

The screenshot shows the RARS 1.6 debugger interface. The assembly code window displays the following instructions:

```

Text Segment
Bkpt Address Code Basic Source
0x00400004 0x00100493 addi x9,x0,1 8: addi s1, zero, 1
0x00400008 0x00400893 addi x17,x0,4 10: li a7, 4
0x0040000c 0x0fc1c0517 auipc x10,0x0000fc10 11: la a0, messl
0x00400010 0xffff450513 addi x10,x10,0xffffffff4
0x00400014 0x00000073 ecall 12: ecall
0x00400018 0x00100893 addi x17,x0,1 14: li a7, 1
0x0040001c 0x00040533 add x10,x8,x0 15: add a0, s0, zero
0x00400020 0x00000073 ecall 16: ecall
0x00400024 0x00400893 addi x17,x0,4 18: li a7, 4

```

The Registers window shows the following register values:

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000004
s1	9	0x00000001
a0	10	0x10010004
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000001
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400024

The Data Segment window shows memory starting at address 0x10010000. The Messages window displays "The sum of 4".

The third ecall

The screenshot shows the RARS 1.6 assembly debugger interface. The assembly window displays the following code:

```

Text Segment
Bkpt Address Code Basic Source
0x00400014 0x00000073 ecall
0x00400018 0x00100893 addi x17,x0,1
0x0040001c 0x00040533 add x10,x8,x0
0x00400020 0x00000073 ecall
0x00400024 0x00400893 addi x17,x0,4
0x0fc10517 auipc x10,0x0000fc10
0x0fe450513 addi x10,x10,0xfffffe4
0x0040002c 0x00000073 ecall
0x00400030 0x00000073 ecall
0x00400034 0x00100893 addi x17,x0,1

```

The instruction at address 0x00400034 is highlighted in yellow. The registers window shows the following state:

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000004
s1	9	0x00000001
a0	10	0x1001000c
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
t3	27	0x00000000
t4	28	0x00000000
t5	29	0x00000000
t6	30	0x00000000
pc	31	0x00400034

The messages window shows the output: "The sum of 4 and".

The fourth ecall

The screenshot shows the RARS 1.6 assembly debugger interface. The assembly window displays the following code:

```

Text Segment
Bkpt Address Code Basic Source
0x00400020 0x00000073 ecall
0x00400024 0x00400893 addi x17,x0,4
0x00400028 0x0fc10517 auipc x10,0x0000fc10
0x0fe450513 addi x10,x10,0xfffffe4
0x0040002c 0x00000073 ecall
0x00400030 0x00000073 ecall
0x00400034 0x00100893 addi x17,x0,1
0x00400038 0x00040533 add x10,x9,x0
0x0040003c 0x00000073 ecall
0x00400040 0x00400893 addi x17,x0,4

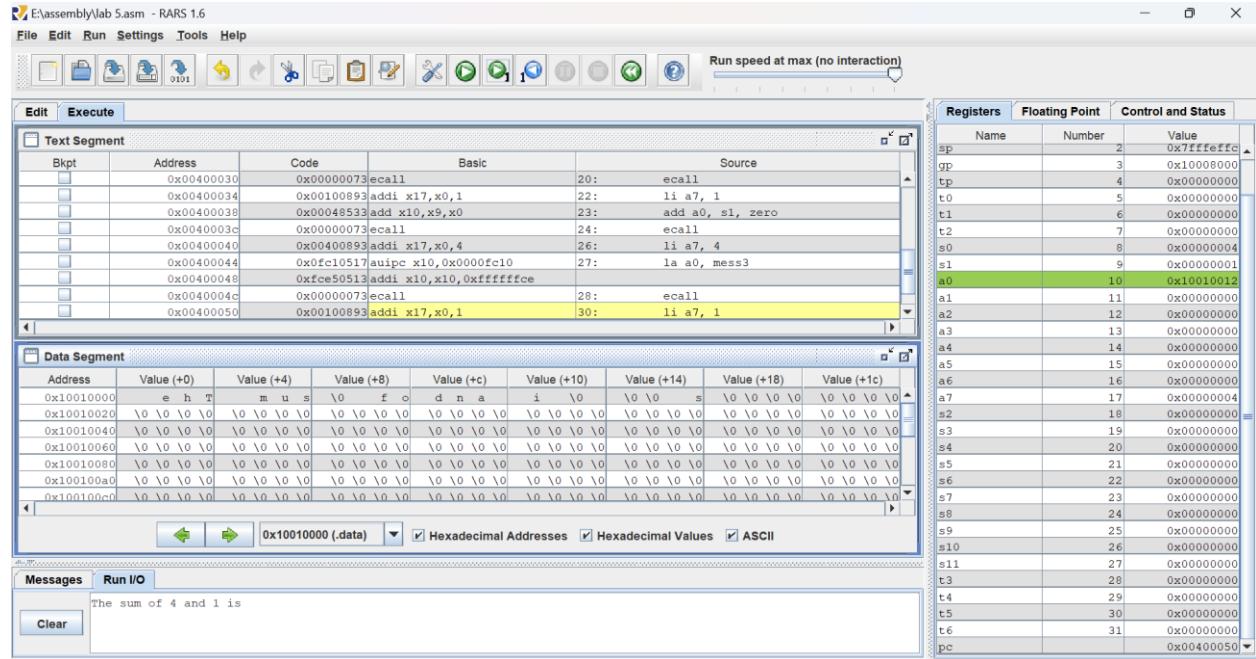
```

The instruction at address 0x00400040 is highlighted in yellow. The registers window shows the following state:

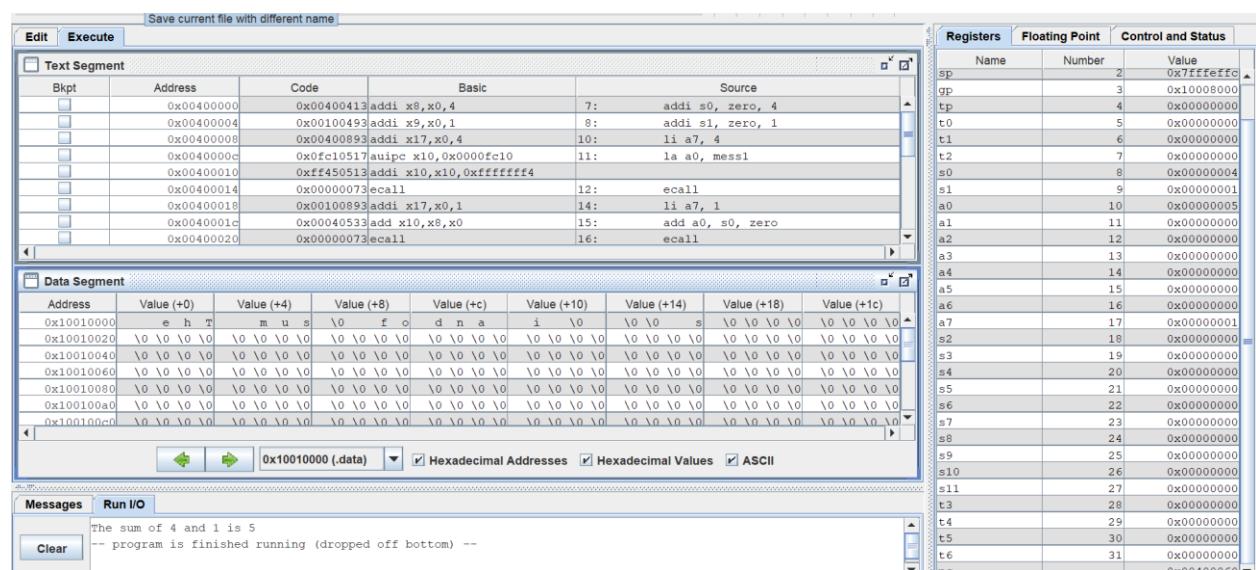
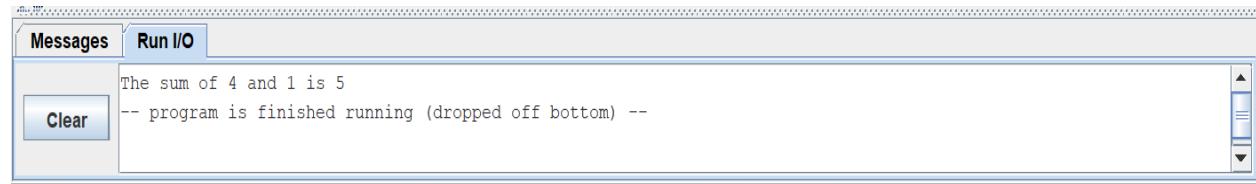
Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000004
s1	9	0x00000001
a0	10	0x00000001
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000001
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
t11	27	0x00000000
t4	28	0x00000000
t5	29	0x00000000
t6	30	0x00000000
pc	31	0x00400040

The messages window shows the output: "The sum of 4 and 1".

The fifth ecall



Finally, after the last ecall, the Run I/O print “The sum of 4 and 1 is 5” correctly as our plan:



Assignment 3: Create a new project to implement the program in Home Assignment 2. Add more instructions to assign a test string for y variable, and implement strcpy function. Compile and upload to simulator. Run and observe the result

Here is the source code with two additional instructions:

We load the address of label x and y into the registers a0 and a1 respectively.

```

1 # Laboratory Exercise 5, Assignment 3
2 .data
3     x: .space 32          # destination string x, empty
4     y: .asciz "Hello"    # source string y
5 .text
6 strcpy:
7     add  s0, zero, zero # s0 = i=0
8     la  a0, x           # additional instruction
9     la  a1, y           # additional instruction
10 L1:
11     add  t1, s0, a1      # t1 = s0 + a1 = i + y[0] = address of y[i]
12     lb  t2, 0(t1)       # t2 = value at t1 = y[i]
13     add  t3, s0, a0      # t3 = s0 + a0 = i + x[0] = address of x[i]
14     sb  t2, 0(t3)       # x[i]= t2 = y[i]
15     beq t2,zero,end_of_strcpy # if y[i]==0, exit
16     addi s0, s0, 1       # s0=s0 + 1 <-> i=i+1
17     j   L1              # next character
18 end_of_strcpy:

```

Line: 9 Column: 35 Show Line Numbers

Before running the program, in Data Segment, we can see that the address of label y is 0x10010020 and y stores the string “Hello”. The address of label x is 0x10010000 and x does not store any thing

Name	Number	Value
sp	2	0x10008000
gp	3	0x00000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
x	9	0x10010000
y	10	0x10010020
a0	11	0x00000000
a1	12	0x00000000
a2	13	0x00000000
a3	14	0x00000000
a4	15	0x00000000
a5	16	0x00000000
a6	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400000

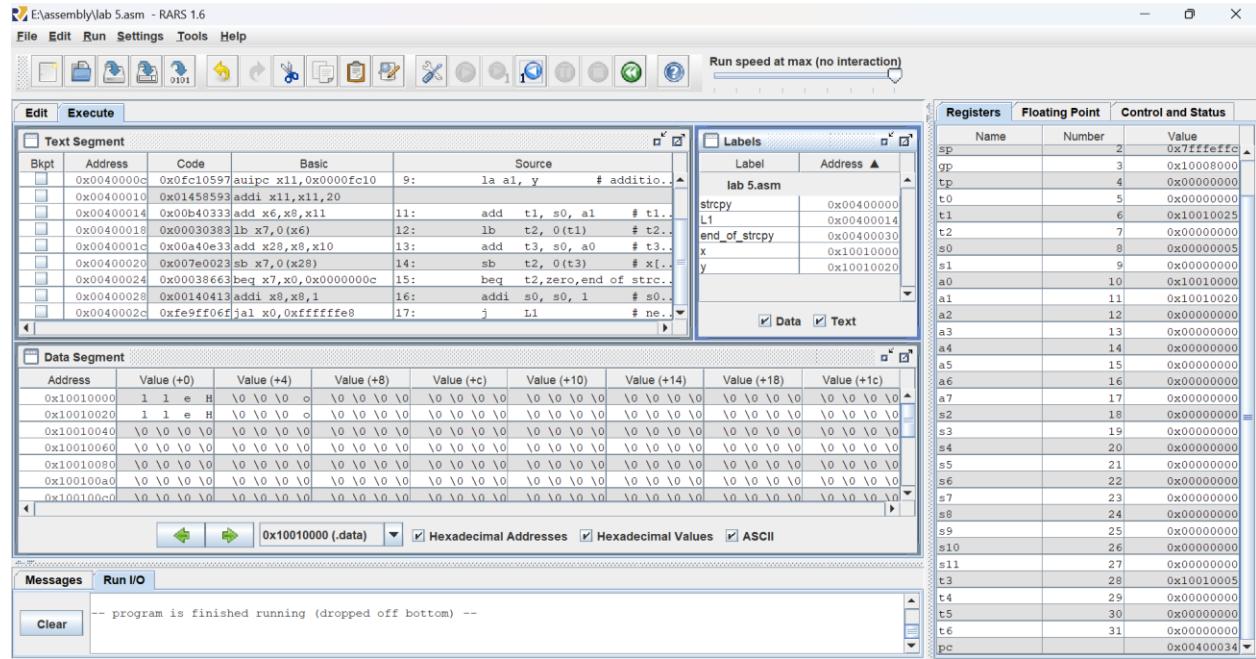
Label	Address
lab 5.asm	0x00400000
strcpy	0x00400014
end_of_strcpy	0x00400030
x	0x10010000
y	0x10010020

Blkpt	Address	Code	Basic	Source
0x00400000	0x000000433	add x8,x0,x0	7:	add s0, zero, zero # s0..
0x00400004	0x0fc10517	auiopc x10,0x0000fc10	8:	la a0, x # additio..
0x00400008	0xffffe50513	addi x10,x10,0xfffff..		
0x0040000c	0x0fc10597	auiopc x11,0x0000fc10	9:	la a1, y # additio..
0x00400010	0x01458593	addi x11,x11,20		
0x00400014	0x00b40333	add x6,x8,x11	11:	add t1, s0, a1 # t1..
0x00400018	0x000030383	lb x7,0(x6)	12:	lb t2, 0(t1) # t2..
0x0040001c	0x00aa40e33	add x28,x8,x10	13:	add t3, s0, a0 # t3..
0x00400020	0x007e0023	sb x7,0(x28)	14:	sb t2, 0(t3) # x[..]

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0	\0	\0	\0	\0	\0	\0	\0
0x10010020	1	1	e	H	\0	\0	\0	\0
0x10010040	\0	\0	\0	\0	\0	\0	\0	\0
0x10010060	\0	\0	\0	\0	\0	\0	\0	\0
0x10010080	\0	\0	\0	\0	\0	\0	\0	\0
0x100100a0	\0	\0	\0	\0	\0	\0	\0	\0
0x100100c0	\0	\0	\0	\0	\0	\0	\0	\0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

After running the program, we can see string "Hello" in the address of label x. It means that x has already copied the string from y.



Assignment 4: Accomplish the Home Assignment 3 with ecall function to get a string from dialog, and show the length to message dialog.

My source code is as below:

```

Edit Execute
lab 5.asm

1 # Laboratory Exercise 5, Assignment 4
2 .data
3     string: .space 50
4     message1: .asciz "Nhap xau: "
5     message2: .asciz "Do dai xau la: "
6
7 .text
8 main:
9 get_string:
10    # TODO Input string from keyboard
11    li a7, 54
12    la a0, message1
13    la a1, string
14    li a2, 50
15    ecall
16 get_length:
17    la a0, string          # a0 = address(string[0])
18    li t0, 0                # t0 = i = 0
19 check_char:
20    add t1, a0, t0          # t1 = a0 + t0 = address(string[0]+i)
21    lb t2, 0(t1)            # t2 = string[i]
22    beq t2, zero, end_of_str # Is null char?

```

Line: 31 Column: 8 Show Line Numbers

Edit Execute

lab 5.asm

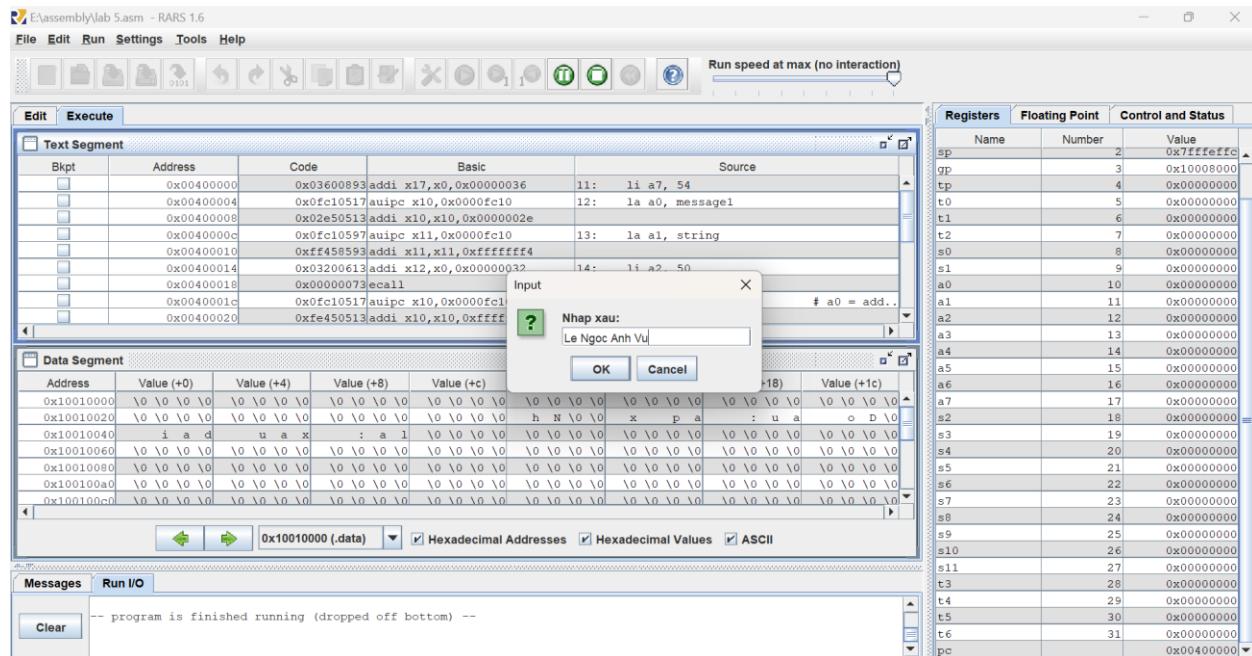
```

12    la a0, message1
13    la a1, string
14    li a2, 50
15    ecall
16    get_length:
17    la a0, string      # a0 = address(string[0])
18    li t0, 0            # t0 = i = 0
19    check_char:
20    add t1, a0, t0      # t1 = a0 + t0 = address(string[0]+i)
21    lb t2, 0(t1)        # t2 = string[i]
22    beq t2, zero, end_of_str # Is null char?
23    addi t0, t0, 1       # t0 = t0 + 1 -> i = i + 1
24    j check_char
25    end_of_str:
26    end_of_get_length:
27    print_length:
28    # TODO print result to console
29    li a7, 56
30    la a0, message2
31    addi a1, t0, -1
32    ecall

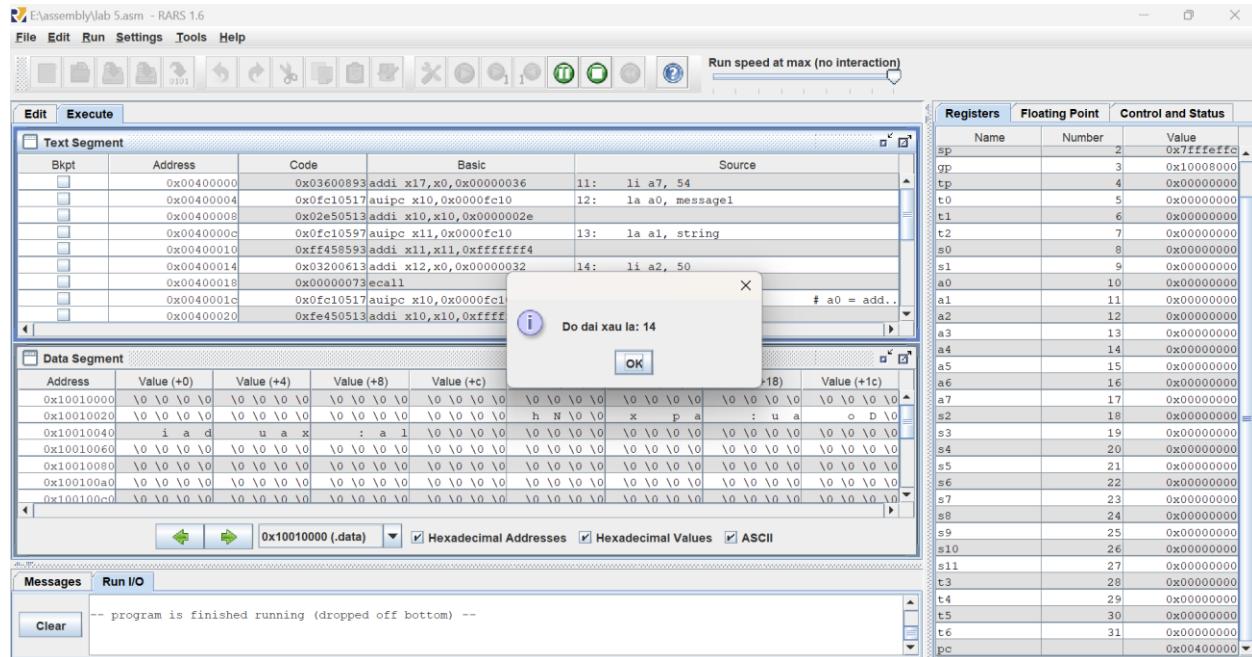
```

Line: 31 Column: 8 Show Line Numbers

We enter a string:



And then, the dialog sends the message about the length of the string.



Assignment 5: Write a program that let user input a string. Input process will be terminated when user press Enter or then length of the string exceed 20 characters. Print the reverse string.

My source code is as below:

```
lab 5.asm
1 # Laboratory Exercise 5, Assignment 5
2 .data
3     string: .space 21
4     message1: .asciz "Enter a string: "
5     message2: .asciz "The reverse string: "
6
7 .text
8 main:
9     li t1, 10
10    li t0, 20
11    li s0, 0      # count the number of characters
12    li a7, 4
13    la a0, message1
14    ecall
15 get_string:
16    li a7, 12
17    ecall
18    breq a0, t1, end
19    addi s0, s0, 1
20    sb a0, 0(sp)
21    addi sp, sp, -1
22    breq s0, t0, endl
Line: 26 Column: 14  Show Line Numbers
```

Edit Execute

lab 5.asm

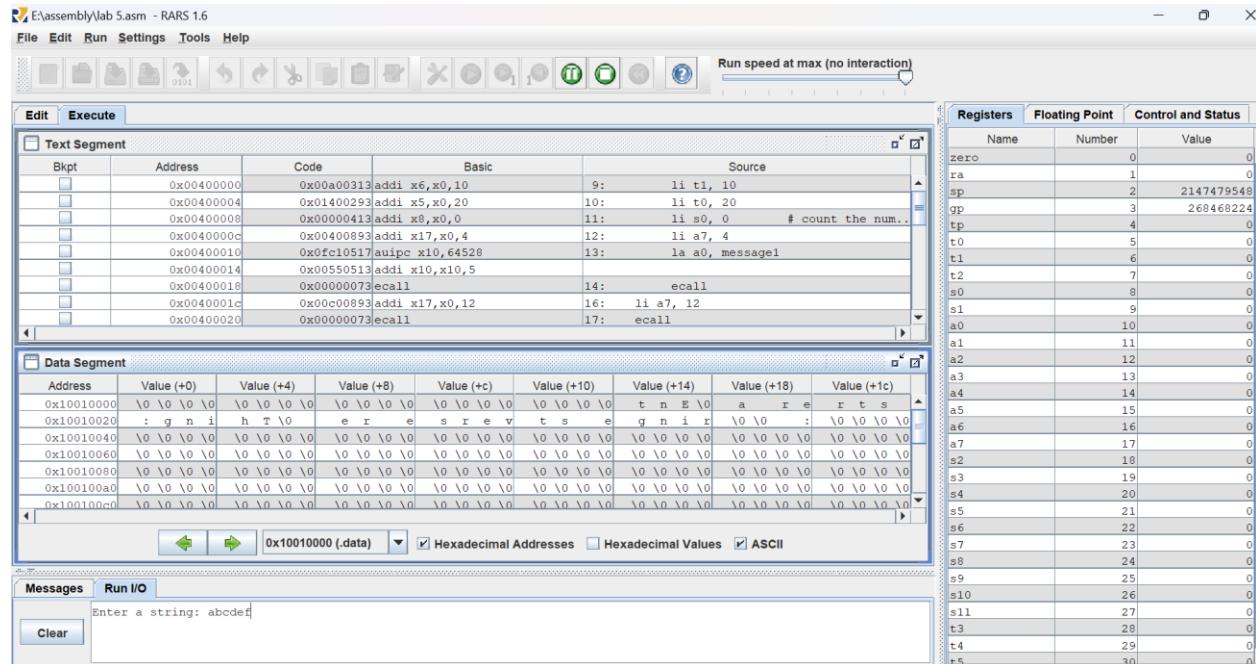
```

21      addi sp, sp, -1
22      beq s0, t0, endl
23      j get_string
24  endl:
25      li a7, 11
26      li a0, '\n'
27      ecall
28 end:
29      li a7, 4
30      la a0, message2
31      ecall
32 reverse:
33      addi sp, sp, 1
34      lb t3, 0(sp)
35      addi s0, s0, -1
36      li a7, 11
37      add a0, t3, zero
38      ecall
39      beqz s0, finish
40      j reverse
41 finish:

```

Line: 26 Column: 14 Show Line Numbers

Case 1: the process input is going to terminated when I press Enter and the program will print the reverse string.



E:\assembly\lab 5.asm - RARS 1.6

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00aa00313	addi x6,x0,10	9: li t1, 10
	0x00400004	0x01400293	addi x5,x0,20	10: li t0, 20
	0x00400008	0x000000413	addi x8,x0,0	11: li s0, 0 # count the num..
	0x0040000c	0x00400893	addi x17,x0,4	12: li a7, 4
	0x00400010	0x0fc10517	auipc x10,64528	13: la a0, message1
	0x00400014	0x00550513	addi x10,x10,5	
	0x00400018	0x00000073	ecall	14: ecall
	0x0040001c	0x00c00893	addi x17,x0,12	16: li a7, 12
	0x00400020	0x00000073	ecall	17: ecall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	t n E \0	a r e	r t s
0x10010020	: g n i h T \0	e r e s r e v t s e	g n i r	\0 \0 :	\0 \0 \0 \0			
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Number	Value
zero	0	0
ra	1	0
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	20
t1	6	10
t2	7	0
s0	8	0
s1	9	0
a0	10	97
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	11
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	97
t4	29	0
t5	30	0

Messages Run I/O

Enter a string: abcdef
The reverse string: fedcba
-- program is finished running (dropped off bottom) --

Case 2: the process input is going to terminated when the length of string exceeds 20 characters.

E:\assembly\lab 5.asm - RARS 1.6

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00aa00313	addi x6,x0,10	9: li t1, 10
	0x00400004	0x01400293	addi x5,x0,20	10: li t0, 20
	0x00400008	0x000000413	addi x8,x0,0	11: li s0, 0 # count the num..
	0x0040000c	0x00400893	addi x17,x0,4	12: li a7, 4
	0x00400010	0x0fc10517	auipc x10,64528	13: la a0, message1
	0x00400014	0x00550513	addi x10,x10,5	
	0x00400018	0x00000073	ecall	14: ecall
	0x0040001c	0x00c00893	addi x17,x0,12	16: li a7, 12
	0x00400020	0x00000073	ecall	17: ecall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	t n E \0	a r e	r t s
0x10010020	: g n i h T \0	e r e s r e v t s e	g n i r	\0 \0 :	\0 \0 \0 \0			
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Number	Value
zero	0	0
ra	1	0
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	20
t1	6	10
t2	7	0
s0	8	0
s1	9	0
a0	10	49
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	11
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	49
t4	29	0
t5	30	0

Messages Run I/O

Enter a string: 12345678901234567890
The reverse string: 09876543210987654321
-- program is finished running (dropped off bottom) --